



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ФУНДАМЕНТАЛЬНЫЕ НАУКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И МАТЕМАТИЧЕСКАЯ  
ФИЗИКА (ФН11)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 02.03.01 МАТЕМАТИКА И КОМПЬЮТЕРНЫЕ  
НАУКИ

## О Т Ч Е Т


по домашнему заданию № 3

В а р и а н т 2 0


Название: *Часть 3.1. Создание модулей. Указатель на функцию.*  
*Часть 3.2. Строки и динамические структуры данных.*

Дисциплина: Информатика

Студент ФН11-12Б  
(Группа)

 19.12.2020 М.Х. Хаписов  
(Подпись, дата) (И.О. Фамилия)

Преподаватель

 Т.Н.Ничушкина  
(Подпись, дата) (И.О. Фамилия)

**20.12.2020**

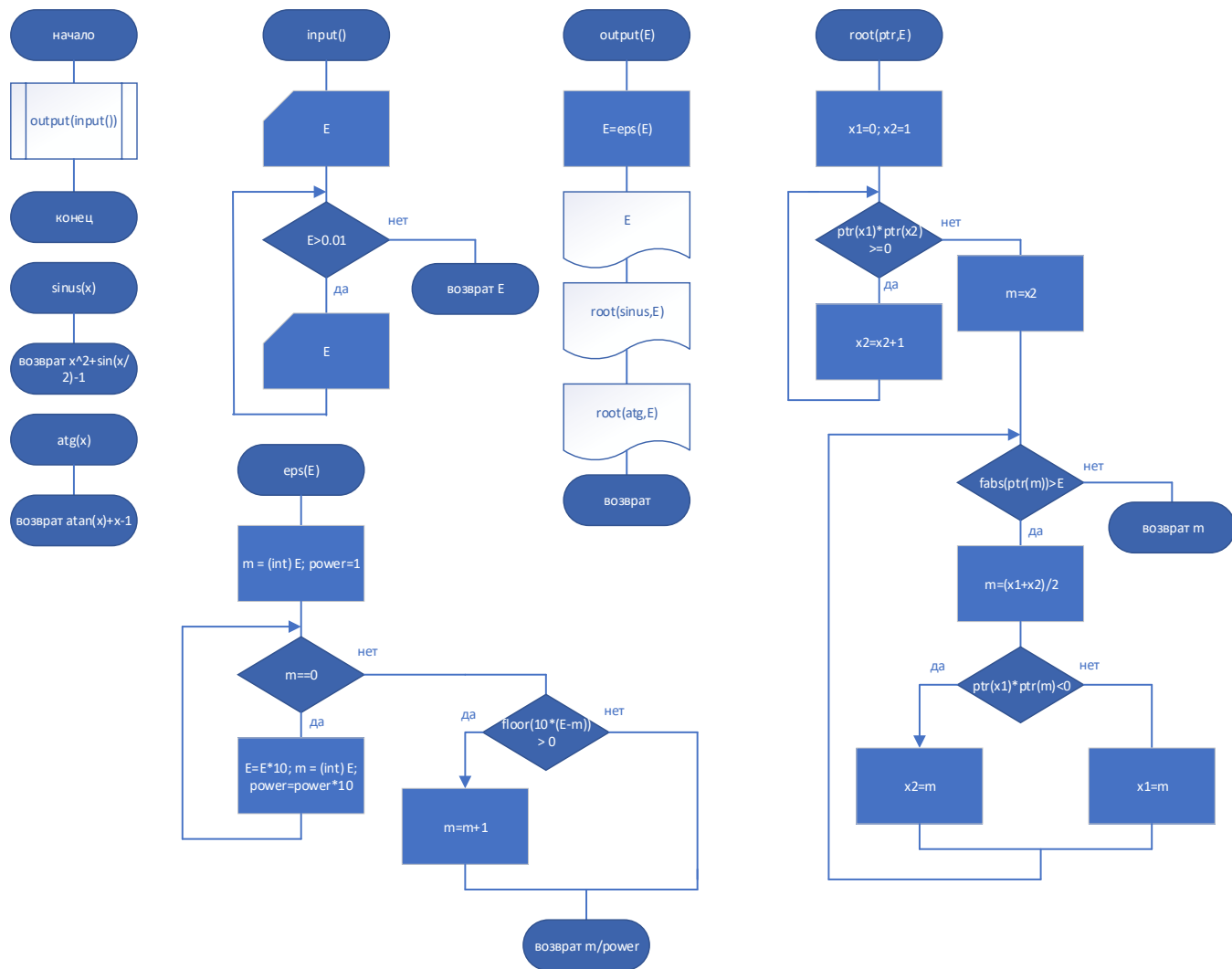
Москва, 2020

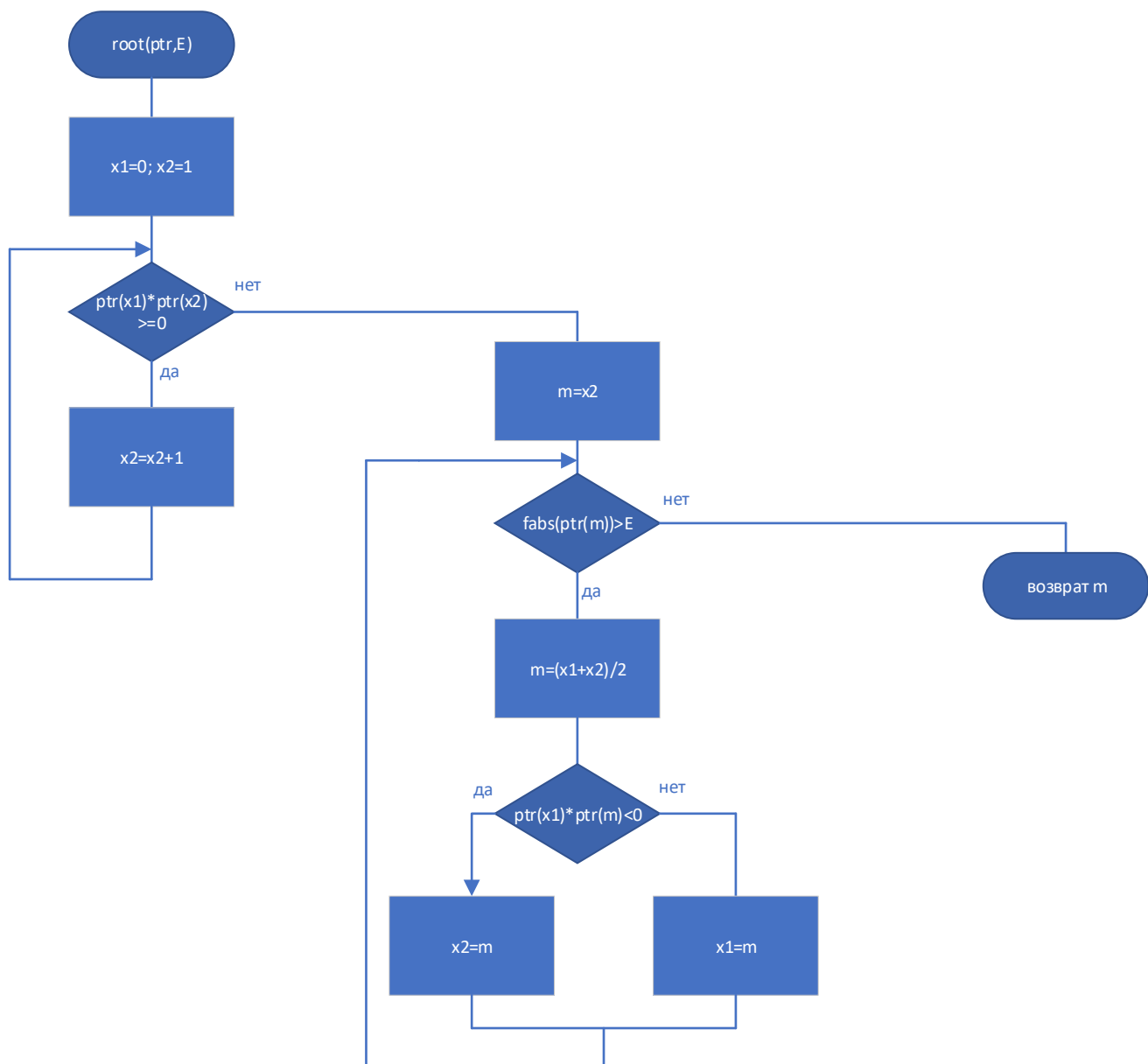
## Создание модулей. Указатель на функцию.

Цель: научиться создавать модули и использовать указатели на функцию

Задача: составить подпрограмму-процедуру KOR отыскания минимального положительного корня уравнения  $f(x)=0$  с точностью 0.01.

### Схема алгоритма





## Текст программы

### Файл main.cpp

```

#include "header.h"
void main() {
    setlocale(0, "rus");
    output(input());
    system("pause");
}

```

### Файл header.h

```

#pragma once
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <cmath>
#include <locale>
using namespace std;
double input();
double eps(double E);
double sinus(double x);
double atg(double x);
double root(double (*ptr)(double), double E);

```

```
void output(double E);
```

### Файл source.cpp

```
#include "header.h"
double eps(double E) { //приведение погрешности к корректному виду
    int m=E;
    int power=1;
    while(m==0) {
        E*=10;
        m=E;
        power*=10;
    }
    if(floor(10*(E-m))>0) m++;
    return 1.0f*m/power;
}
double sinus(double x) {
    return x*x+sin(x/2)-1;
}
double atg(double x) {
    return atan(x)+x-1;
}
double root(double (*ptr)(double),double E) {
    double x1=0,x2=1;
    while (ptr(x1)*ptr(x2)>=0) x2++;
    double m=x2;
    while (fabs(ptr(m))>E) {
        m=(x1+x2)/2;
        if (ptr(x1)*ptr(m)<0) x2=m;
        else x1=m;
    }
    return m;
}
double input() {
    double E;
    cout<<"Введите погрешность вычислений\n";
    cin>>E;
    while(E>0.01) {
        cout<<"Погрешность слишком мала!\n";
        cin>>E;
    }
    return E;
}
void output(double E) {
    E=eps(E);
    double temp=E;
    int n=0;
    while(temp<1) {
        temp*=10;
        n++;
    }
    if(temp>9) n--;
    printf("Погрешность вычислений равна %g\n",E);
    cout<<setprecision(n)<<"Корень уравнения x^2+sin(x/2)=1:"
    "<<root(sinus,E)<<"\n";
    cout<<setprecision(n)<<"Корень уравнения arctg(x)+x=1:"
    "<<root(atg,E)<<"\n";
}
```

## Тестирование

```
C:\Users\Knigan\source\repos\Project3\Debug\Project3.exe
Введите погрешность вычислений
0.01
Погрешность вычислений равна 0,01
Корень уравнения  $x^2 + \sin(x/2) = 1$ : 0.78
Корень уравнения  $\arctg(x) + x = 1$ : 0.52
Для продолжения нажмите любую клавишу . . .
```

```
C:\Users\Knigan\source\repos\Project3\Debug\Project3.exe
Введите погрешность вычислений
0.062
Погрешность вычислений равна 0,07
Корень уравнения  $x^2 + \sin(x/2) = 1$ : 0.81
Корень уравнения  $\arctg(x) + x = 1$ : 0.5
Для продолжения нажмите любую клавишу . . .
```

```
C:\Users\Knigan\source\repos\Project3\Debug\Project3.exe
Введите погрешность вычислений
0.001
Погрешность вычислений равна 0,001
Корень уравнения  $x^2 + \sin(x/2) = 1$ : 0.785
Корень уравнения  $\arctg(x) + x = 1$ : 0.521
Для продолжения нажмите любую клавишу . . .
```

```
C:\Users\Knigan\source\repos\Project3\Debug\Project3.exe
Введите погрешность вычислений
0.0089
Погрешность вычислений равна 0,009
Корень уравнения  $x^2 + \sin(x/2) = 1$ : 0.781
Корень уравнения  $\arctg(x) + x = 1$ : 0.516
Для продолжения нажмите любую клавишу . . .
```

```
C:\Users\Knigan\source\repos\Project3\Debug\Project3.exe
Введите погрешность вычислений
0.0001
Погрешность вычислений равна 0,0001
Корень уравнения  $x^2 + \sin(x/2) = 1$ : 0.7856
Корень уравнения  $\arctg(x) + x = 1$ : 0.5203
Для продолжения нажмите любую клавишу . . .
```

```
C:\Users\Knigan\source\repos\Project3\Debug\Project3.exe
Введите погрешность вычислений
0.00085
Погрешность вычислений равна 0,0009
Корень уравнения  $x^2 + \sin(x/2) = 1$ : 0.7856
Корень уравнения  $\arctg(x) + x = 1$ : 0.5205
Для продолжения нажмите любую клавишу . . .
```

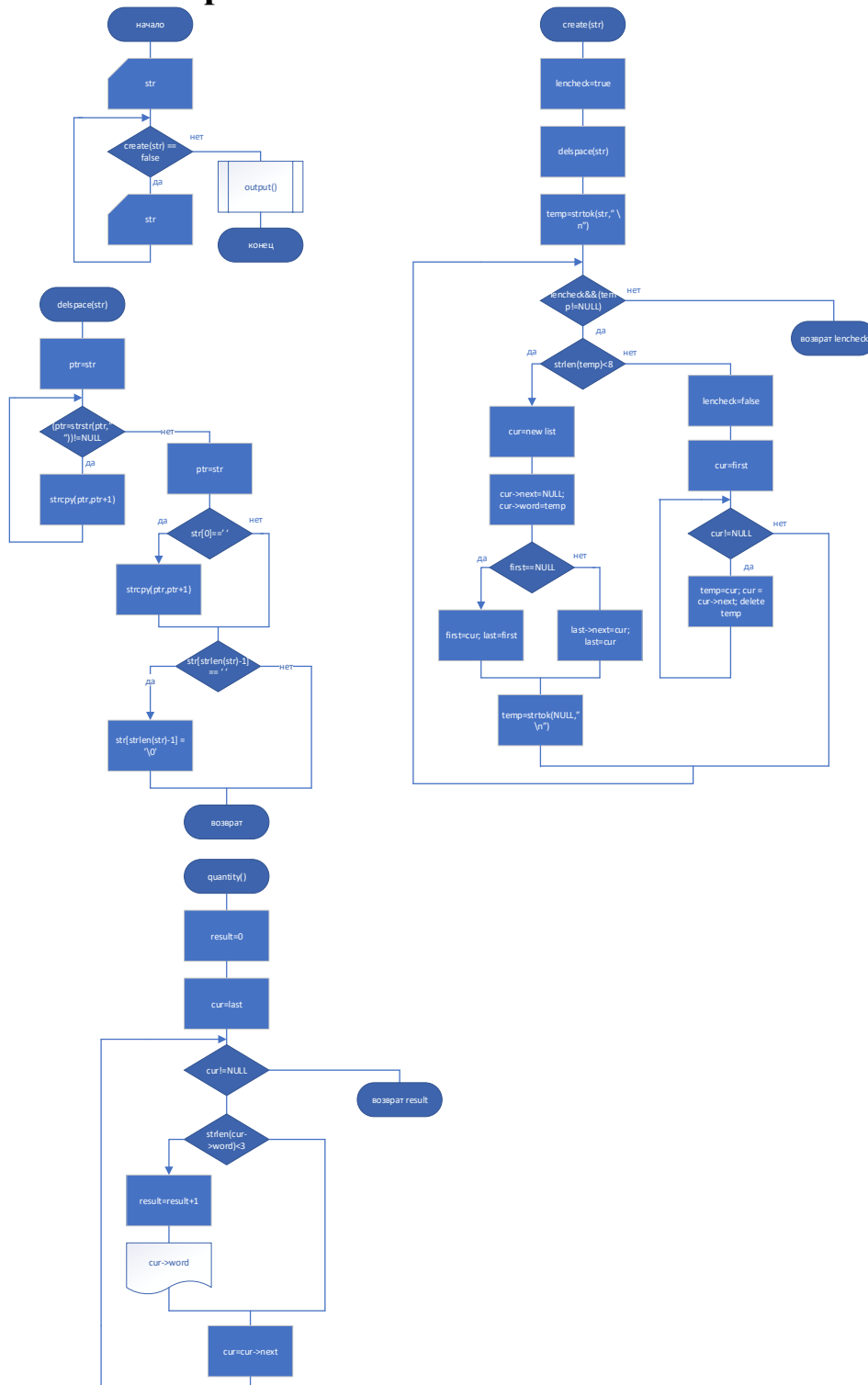
**Вывод:** я научился создавать модули и использовать указатели на функцию

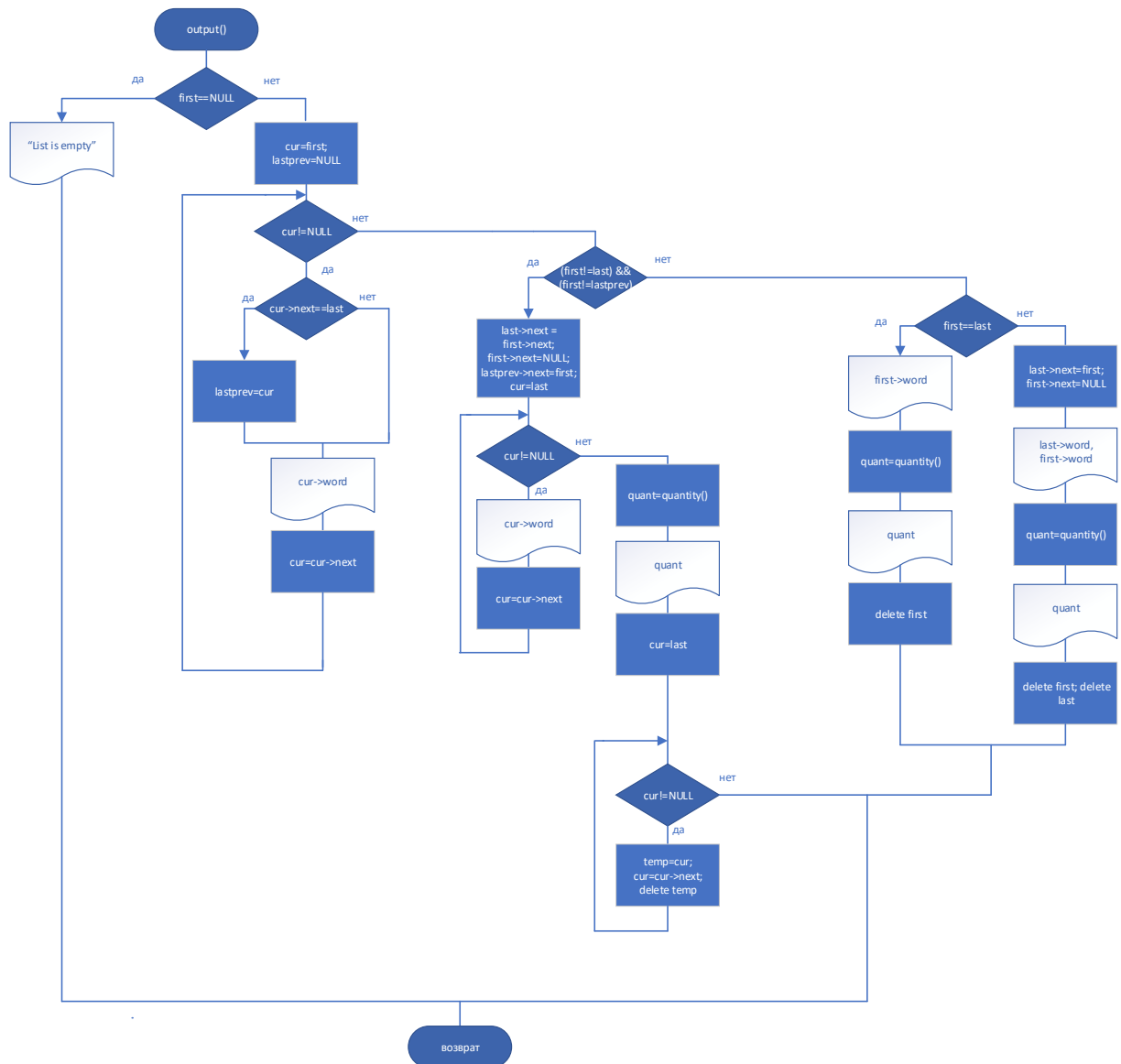
# Строки и динамические структуры данных

**Цель:** научиться использовать строки и динамические структуры данных

**Задача:** составить программу, которая вводит строку, организует из слов строки длиной не более 7 однонаправленный список, в сформированном списке меняет местами первое и последнее слово, подсчитывает слова, длина которых меньше 3, выводит на экран результаты работы

## Схема алгоритма





## Текст программы

### Файл main.cpp

```

#include "header.h"
void main() {
    char str[81];
    puts("Input string");
    gets_s(str);
    while(create(str)==false) {
        puts("Length of every word must be less than 8");
        gets_s(str);
    }
    output();
    system("pause");
}

```

### Файл header.h

```

#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void delspace(char *str);

```

```

bool create(char *str);
int quantity();
void output();

```

## Файл source.cpp

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio>
#include <stdlib>
#include <cstring>
struct list {
    char *word;
    list *next;
};
list *first=NULL,*last=NULL;
void delspace(char *str) {
    char *ptr;
    ptr=str;
    while((ptr=strstr(ptr, " "))!=NULL) strcpy(ptr,ptr+1);
    ptr=str;
    if(str[0]==' ') strcpy(ptr,ptr+1);
    if(str[strlen(str)-1]==' ') str[strlen(str)-1]='\0';
}
bool create(char *str) {
    bool lencheck=true;
    delspace(str);
    char *temp=strtok(str, " \n");
    while(lencheck && (temp!=NULL)) {
        if(strlen(temp)<8) {
            list *cur=new list;
            cur->next=NULL;
            cur->word=temp;
            if(first==NULL) {
                first=cur;
                last=first;
            } else {
                last->next=cur;
                last=cur;
            }
            temp=strtok(NULL, " \n");
        } else {
            lencheck=false;
            list *cur=first;
            while(cur!=NULL) {
                list *temp=cur;
                cur=cur->next;
                delete temp;
            }
        }
    }
    return lencheck;
}
int quantity() {
    int result=0;
    puts("Words of length less than 3:");
    list *cur=last;
    while(cur!=NULL) {
        if(strlen(cur->word)<3) {
            result++;
            printf("%s ",cur->word);
        }
        cur=cur->next;
    }
    puts("");
}

```



```

        return result;
    }
}

void output() {
    if (first==NULL) puts("List is empty");
    else {
        puts("Initial string:");
        list *cur=first,*lastprev=NULL;
        while(cur!=NULL) {
            if(cur->next==last) lastprev=cur;
            printf("%s ",cur->word);
            cur=cur->next;
        }
        puts("");
        puts("Transformed string:");
        if ((first!=last)&&(first!=lastprev)) {
            last->next=first->next;
            first->next=NULL;
            lastprev->next=first;
            cur=last;
            while(cur!=NULL) {
                printf("%s ",cur->word);
                cur=cur->next;
            }
            puts("");
            int quant=quantity();
            printf("Quantity of words of length less than 3 is
%d\n",quant);

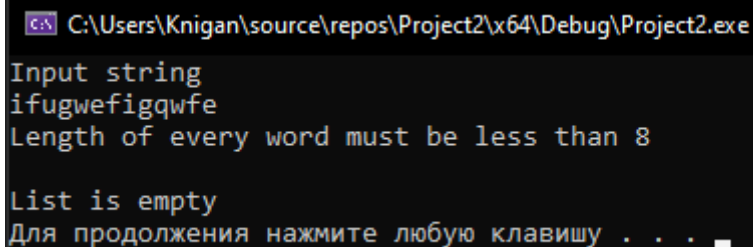
            cur=last;
            while(cur!=NULL) {
                list *temp=cur;
                cur=cur->next;
                delete temp;
            }
        } else {
            if (first==last) {
                puts(first->word);
                int quant=quantity();
                printf("Quantity of words of length less than 3 is
%d\n",quant);

                delete first;
            } else {
                last->next=first; first->next=NULL;
                printf("%s %s\n",last->word,first->word);
                int quant=quantity();
                printf("Quantity of words of length less than 3 is
%d\n",quant);

                delete first; delete last;
            }
        }
    }
}
}

```

## Тестирование



```

C:\Users\Knigan\source\repos\Project2\x64\Debug\Project2.exe
Input string
ifugwefigqwfe
Length of every word must be less than 8

List is empty
Для продолжения нажмите любую клавишу . . .

```

C:\Users\Knigan\source\repos\Project2\x64\Debug\Project2.exe

```
Input string
sdfasdf cvs, a. dg
Initial string:
sdfasdf cvs, a. dg
Transformed string:
dg cvs, a. sdfasdf
Words of length less than 3:
dg a.
Quantity of words of length less than 3 is 2
Для продолжения нажмите любую клавишу . . .
```

C:\Users\Knigan\source\repos\Project2\x64\Debug\Project2.exe

```
Input string
weiofgwefig dffaf mk
Length of every word must be less than 8
fdusdf ;qwoe a
Initial string:
fdusdf ;qwoe a
Transformed string:
a ;qwoe fdusdf
Words of length less than 3:
a
Quantity of words of length less than 3 is 1
Для продолжения нажмите любую клавишу . . .
```

C:\Users\Knigan\source\repos\Project2\x64\Debug\Project2.exe

```
Input string
fsd f  ah sdfh a    dsf
Initial string:
fsd f ah sdfh a dsf
Transformed string:
dsf f ah sdfh a fsd
Words of length less than 3:
f a
Quantity of words of length less than 3 is 2
Для продолжения нажмите любую клавишу . . .
```

C:\Users\Knigan\source\repos\Project2\x64\Debug\Project2.exe

```
Input string
a ba
Initial string:
a ba
Transformed string:
ba a
Words of length less than 3:
ba a
Quantity of words of length less than 3 is 2
Для продолжения нажмите любую клавишу . . .
```

```
C:\Users\Knigan\source\repos\Project2\x64\Debug\Project2.exe
Input string
[]
Initial string:
[]
Transformed string:
[]
Words of length less than 3:
[]
Quantity of words of length less than 3 is 1
Для продолжения нажмите любую клавишу . . .
```

**Вывод:** я научился использовать строки и динамические структуры данных

### Ответы на контрольные вопросы

1. Дайте определение подпрограммы. (Ответ: подпрограмма – это относительно самостоятельный фрагмент программы, соответствующим образом оформленный и снабженный именем)
2. Что такое модуль, как его создать с помощью средств C++? (Ответ: модуль – программа, использующая не только стандартные, но и пользовательские библиотеки. Модуль C++ обычно включает в себя 2 файла: заголовочный файл с расширением «.h» и файл реализации с расширением «.cpp»)
3. Укажите особенности подключения модулей в среде Visual Studio 2008.

При создании первый (главный) файл проекта уже содержит заготовку основной функции программы – функции `main()`. Для создания файлов модуля и добавления их к проекту можно использовать два пути:

- 1) Вновь вызвать многошаговый Мастер заготовок. Это делается с использованием команды меню `File/New`. Выполнение этой команды при открытом проекте вызовет открытие окна Мастера заготовок на вкладке `Files`, на которой необходимо выбрать тип файла, добавляемого к проекту и задать его имя.
- 2) Использовать команду меню `Project/Add New Items`. Выполнение этой команды также вызовет открытие окна Мастера заготовок, с помощью которого необходимо выбрать тип файла, добавляемого к проекту и задать его имя.

4. Что такое указатель на функцию? Приведите примеры описания и применения.

Функция характеризуется типом возвращаемого значения, именем и сигнатурой. Имя функции – это указатель, хранящий ее адрес, который может быть присвоен другому указателю. Однако при объявлении для нового указателя должен быть задан тот же тип, что и возвращаемое функцией значение, и тот же список параметров с точностью до типов формальных параметров (имена параметров могут различаться).

Указатель на функцию объявляется следующим образом:

**<Тип функции>(\* <Имя>)(<Спецификация параметров>);**

5. Какие особенности передачи в подпрограмму указателей, как параметров?

```
float (* funuk)(float); // указатель на функцию
float integral(float(*funuk)(float),float a,float b,float eps)
{
    int i,n=5,k=0; float s1,s2=1.0e+10,x,d;
    d=(b-a)/n;
    do
    {
        s1=s2; s2=0;n=n*2; d=d/2;
        x=a; k++;
        for(i=1;i<=n;i++)
        {
            s2=s2+funuk(x);
            x=x+d;
        }
        s2=s2*d;
    }
    while(fabs(s2-s1)>eps);
    return s2;
}
```

6. Где и когда используются целесообразно использовать указатели на функции? (Указатели на функцию используются с целью сделать подпрограммы, использующие эти указатели, универсальными)

7. Что такое «список» в программировании? В каких случаях используется эта конструкция? (Список – это структура, при организации которой использованы указатели, содержащие адреса следующих элементов)

8. Какие типы списковых структур вы знаете?

По структуре списки бывают линейными, древовидными и сетевыми. На линейных списках обычно реализуют разные дисциплины обслуживания:

- очередь – дисциплина обработки элементов данных, в которой добавление элементов выполняется в конец, а удаление – из начала;
- стек – дисциплина обработки элементов данных, в которой добавление и удаление элементов осуществляется с одной стороны;
- дек – дисциплина обработки элементов данных, в которой добавление и удаление элементов может выполняться с двух сторон.

9. Что такое указатели и как они объявляются?

Указатель – это переменная, значением которой является адрес ячейки памяти. Указатель объявляется следующим образом:

<Тип указателя>(\*<Имя>)[=<значение>]

10. Как описывается элемент списка?

Элемент односвязного списка:

```
struct element // тип элемента
{
    char name[16]; // информационное поле 1
    char telefon[7]; // информационное поле 2
    element *p; // адрес следующего элемента
};
```

Элемент двусвязного списка:

```
struct element // тип элемента
{
```

```
char name[16]; // информационное поле 1
char telefon[7]; // информационное поле 2
element *prev; // адресное поле <<предыдущий>>
element *next; // адресное поле <<следующий>>
};
```

11. Какие варианты возможны для добавления элемента к списку? (Ответ: добавление элемента к пустому списку, добавление элемента перед первым элементом списка, добавление элемента перед заданным, добавление элемента в конец списка)

12. Как отлаживают программы, содержащие обработку списков? (Ответ: такие программы отлаживаются таким образом, чтобы протестировать работу программы во всех возможных критических ситуациях)

Вывод: я научился создавать модули, использовать указатели на функцию, научился использовать строки и динамические структуры данных.